Policy-Based Security Modelling and Enforcement Approach for Emerging Embedded Architectures

Matthew Hagan, Fahad Siddiqui, Sakir Sezer,
The Centre for Secure Information Systems (CSIT), Queen's University Belfast
Belfast, United Kingdom
m.hagan, f.siddiqui, s.sezer@qub.ac.uk

***Abstract*—Complex embedded systems often contain hard to find vulnerabilities which, when exploited, have potential to cause severe damage to the operating environment and the user. Given that threats and vulnerabilities can exist within any layer of the complex eco-system, OEMs face a major challenge to ensure security throughout the device life-cycle To lower the potential risk and damage that vulnerabilities may cause, OEMs typically perform application threat analysis and security modelling. This process typically provides a high level guideline to solving security problems which can then be implemented during design and development. However, this concept presents issues where new threats or unknown vulnerability has been discovered.**

To address this issue, we propose a policy-based security modelling approach, which utilises a configurable policy engine to apply new policies that counter serious threats. By utilising this approach, the traditional security modelling approaches can be enhanced and the consequences of a new threat greatly reduced.

We present a realistic use case of connected car, applying several attack scenarios. By utilising STRIDE threat modelling and DREAD risk assessment model, adequate policies are derived to protect the car assets. This approach poses advantages over the standard approach, allowing a policy update to counter a new threat, which may have otherwise required a product redesign to alleviate the issue under the traditional approach.

***Index Terms*—Secure by design, Security Modelling, Policy, Embedded Security, STRIDE, Threat Modelling.**

I. Introduction

The emergence of embedded devices within industrial and consumer products hold great potential in terms of productivity, quality control, supply chain efficiency and overall business operations. The embedded market has grown significantly, with a massive range of products available to the consumer, enterprise and industry. Estimates from ARM predict that emerging connected embedded devices will proliferate to a trillion devices by 2035 [1]. Most of these devices will have inter-connectivity that exposes them to a myriad of security risks and attacks, often derived from methods used to attack conventional IT systems.

The majority of these security mechanisms rely on well established software security practices. Reported attacks and discovered vulnerabilities have shown that the device security are often implemented poorly, or in some cases have not been considered, leading to attacks such as the Mirai botnet [2], [3]. From an embedded architecture point-of-view, a major reason for this is the need for varying levels of security, cannot be defined generically due to being highly dependent on the deployment scenario which could be based on consumer, transport, medical and industrial control applications. As a consequence, devices have been found vulnerable, with the same flaws affecting different use cases, leading to attacks affecting privacy, device integrity, and risking loss of sensitive information [4], [5].

The widely practised approach for devising the security architecture of a device is to conduct use case-based *Application threat modelling*. This process assesses the use case, deployment scenarios and device functionalities, identifying the device assets that an adversary may target [6], [7]. This process is defined in a way that complements the device development life-cycle. The end-product of this process is a *security model*, a technical document that provides security guidelines specific to that use case. The designers then follow these security guidelines, where possible, to achieve the desired application security. The device is therefore created to be resistant against only threats conisdered during the modelling process. In practice, threat modelling can be compromised by corporate practices, or invalidated by newly discovered threats that were not considered during the development phase threat modelling process. Changes to a device security and threat profile requires updating the security and threat model to ensure device security compliance. Technically, this may require re-designing of hardware and software implementation, which in many cases is neither cost effective nor technically feasible [8].

Deployment of these powerful, yet generic embedded devices in diverse and critical applications poses new challenges for the semiconductor and hardware platform manufacturers, who must provide security assurance [9] throughout the product life-cycle, from product idea to decommission [8]. International regulators have implemented further requirements that quantify the security of devices and specify *security ratings* [10]. These security and privacy laws will be enforced across the landscape of embedded devices, meaning that OEMs and device manufacturers must seek security assurance from semiconductor/platform manufacturers for compliance.

To investigate these research challenges, we present a policy-based security modelling and enforcement approach, that can be incorporated into embedded architectures to address some challenges faced by the emerging embedded applications. Section II will discuss existing security and threat modelling practices. Section III will review relevant related-work on security modelling, observing strengths and short-

comings. Section V will introduce our proposed policy-based approach, followed by a connected car use case. Section VI will discuss our findings and propose future work in the area.

II. Background

The US government has performed extensive work into implementing security policy for IT systems with particular focus on *Mandatory Access Control* (MAC) and *Discretionary Access Control* (DAC) [11], [12]. Later, the National Security Agency (NSA) [13] extended it by developing the *Security Enhanced Linux* (SELinux) platform [14] that can enforce access control policies at kernel level. Similarly, ARM [7] has published a security model specifications known as the *Platform Security Architecture* (PSA), which aims to simplify and enhance the security evaluation process by providing guidelines and steps for creating a security model for connected embedded devices.

Application threat modelling is a process of assessing the security requirements of an application use case. The detailed block diagram of design and development stages along with the tasks necessary for Application Threat Modelling in Fig. 1. The following are the key steps involved:

- Risk assessment: This step is to gain understanding of an application use case, decomposing it to identify interactions of different internal and external entities and their associated risks. They are documented as the system's *security requirements*.
- Identify Assets: This identifies items of value that should be protected. Dependent assets within the use case which can also be examined from a data flow perspective.
- Entry Points: They are interfaces that exposes critical assets to the attacker, and can be used to interact with the system or application.
- Threat Identification: It involves identification of potential vulnerabilities within the system that can be exploitable by an attacker with or without knowing the architecture and implementation details. These threats are then categorised using categorisation methodologies such as STRIDE, as used in this paper. This step is technically documented as the system *threat model*.
- Threat Rating: The identified threat are prioritise and quantified based on their likelihood, risk and potential damage to the assets or the system infrastructure. The evaluated security risk for each threat is quantified by assigning a value using risk-based models such as DREAD.
- Determine countermeasure: It involves defining a countermeasure against each of the identified threats depending on their prioritised risk ratings. Countermeasures can be deployed in hardware and software. Smaller threats could be catered using best security practises.

At this stage, a detailed technical guidance document is available for both the hardware and software developers and their implementation must comply to these requirements which are then used for *Secure application testing* phase. These guidelines direct developers when navigating security sensitive areas of the device. However, issues are likely to occur where the programmer has less control over certain aspects, such as third-party IP blocks. Since the OEM has less influence over the design process for these components, they are constrained in terms of how they mitigate potential vulnerabilities.

However, in reality, threats may be overlooked or remain undiscovered during product development. As recognised by [15], threat modelling is largely a theoretical analysis task, dependent on human insights to identify threats. With extensive use of third party IP, identifying threats within largely unknown components is often not possible. Therefore, when a new threat emerges, the existing security model may be invalidated, due to the change in risk profile of the device, undermining the existing security model.

III. Related-work

This section will survey relevant applications of threat modelling to device and system environments.

Dorsemaine *et al.* [16] investigated potential attacks against corporate infrastructures that leveraged compromised IoT devices. They observed IoT from a layered perspective (objects, transport, storage, interfaces), performing an example risk analysis against the featuresets of a range of connected IoT thermostats. This work did not perform a prescriptive threat model approach such as STRIDE. In addition this work used guidelines to recommend attack mitigations.

Khan *et al.* [17] demonstrated the application of STRIDE to cyber physical systems to identify interdependencies and security threats. STRIDE was found to effectively characterise system threats while being lightweight. This work however did not discuss countermeasures and mitigations. [18] further used the STRIDE and DREAD models to mobile health applications. Their work proposed mitigation strategies to assist modifications to the development process.

Tan *et al.* presented a decentralised system-level security approach, using resource isolation as the security foundation of the embedded architecture [19]. Resource isolation is defined in the form of access control policies classified into base, owner and shared permissions. These access control policies enable a hierarchical security profile for each embedded resource, with the device manufacturer, OEMs and users given restricted control to manage it. They presented an in-depth example scenario of home automation, covering security requirements and potential threats. However, their threat modelling approach is not holistic, lacking quantification of threat analysis, based on well-established models such as STRIDE and DREAD. Akatyev etal. presented work on investigating current and future threats to near future inter-connected systems [20]. They described such systems as user-centric IoT, based on offered services to the user and their direct impact through physical environment. Contrary to Tan *et al.* they have adopted a holistic approach, categorising identified attacks and related threats using widely adopted STRIDE and DREAD models. They facilitate quantification of the combined likelihood and impact of each threat, giving improved visibility, to prioritise design effort. They performed a detailed investigation of a real-

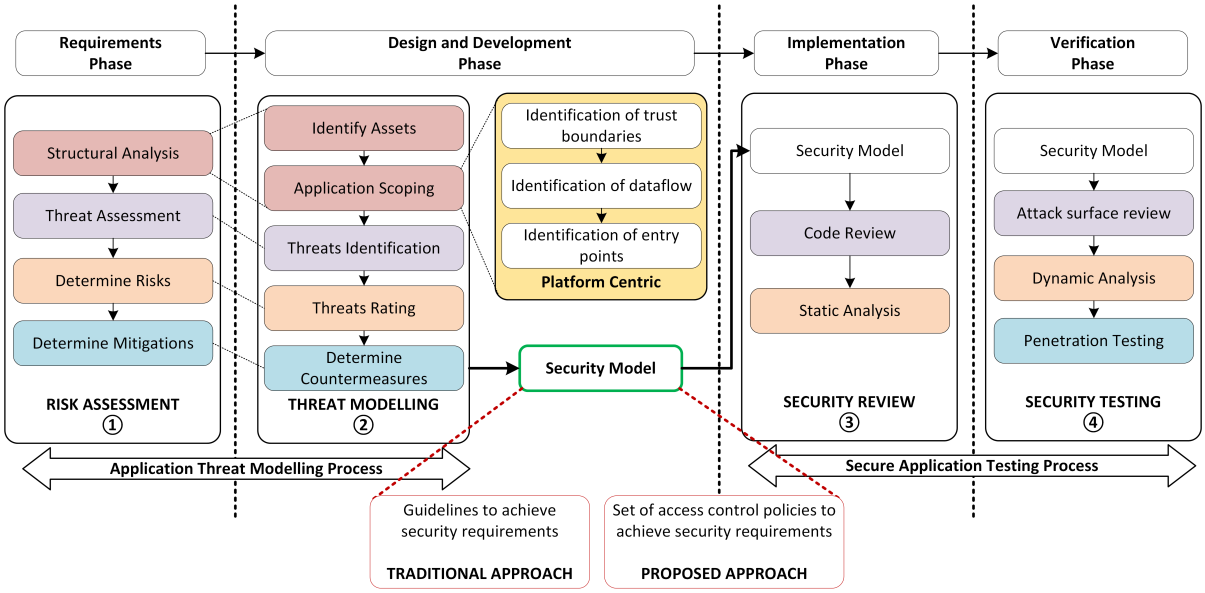

Fig. 1. Step-wise illustration of secure product development life-cycle covering both *application threat modelling* and *secure application testing* process. The device security model is a task that serves as a bridge between these two processes that can be defined as access control policies.

world case study, however, implementation and incorporation into existing design practises was not covered.

This paper aims to extend the policy driven approach presented by Tan *et al.* [19], by presenting a policy-based security modelling design flow incorporating its output into the existing application threat modelling practises and design flow. In addition, this proposed approach compliments the work of Akatyev *et al.* [20] and Siddiqui *et al.* [21] by performing detailed threat modelling of a realistic connected car application scenario and proposing ways to enforce driven security policies using hardware and software approaches.

IV. Policy-based Security Modelling Approach

As discussed in Section II, the traditional approach at this stage is to define technical guidance, which is then used within the implementation. These guidelines direct developers when navigating security sensitive areas of the device.

Our contribution to this area is to propose a policy-based security model that can be tailored to the security needs of the use case, providing a flexible security model that is manageable and adaptable during the device life-cycle. The enforcement of policies can be carried out by a software component, such as SELinux, or by an external hardware entity. In the event of an attempted re-purposing or exploit, a hardware-based security mechanism can mitigate the intrusion to protect the device. The policy enforcement engine comprises the following features:

- *Hardware*: Monitoring system-level communication between data and memory peripherals, enforcing the defined security policies for each slave.
- *Software*: Checking application permission boundaries and identifying anomalous behaviour.

By utilising policies to enforce these requirements, the OEM does not have to rely on third party vendors' security assurances. Enforcing policies, derived from threat modelling of the system, assets and attack scenarios, can ensure the device operates as intended by the OEM. Development costs may also be lowered for the hardware vendor if a generic platform can be distributed to multiple users, who can derive their own security policies based on the security requirements and level of security desired using the familiar *Application Threat Modelling* approach presented in Fig. 1 and enforce them. Finally, should the security requirements of the device change after production, for example, a new vulnerability is discovered, the OEM can distribute a policy definition update.

V. Case Study: A connected car system

We have envisaged a *connected car* scenario for the purpose of presenting a threat analysis and security modelling process, considering both the standard, guideline-based approach and our proposed policy-based approach.

A connected car features a variety of complex, interconnected systems of differing levels of criticality, including vehicle controls, sensors-based critical safety, infotainment, telematics and cellular network access [22]. Various components comprise part of multiple systems of differing criticality. For example, cellular connectivity within the vehicle may be used for entertainment or navigation purposes by the user, for telemetry upload by the vehicle, to update firmware by the manufacturer, or used to notify emergency services of an accident or to deactivate to vehicle in case of reported theft.

Controller Area Network (CAN) bus is a de-facto bus communication protocol used in cars and standardised as ISO 11898 [23]. CAN is a differential 2-wire, multi-drop and multi-master serial bus protocol that allows communication

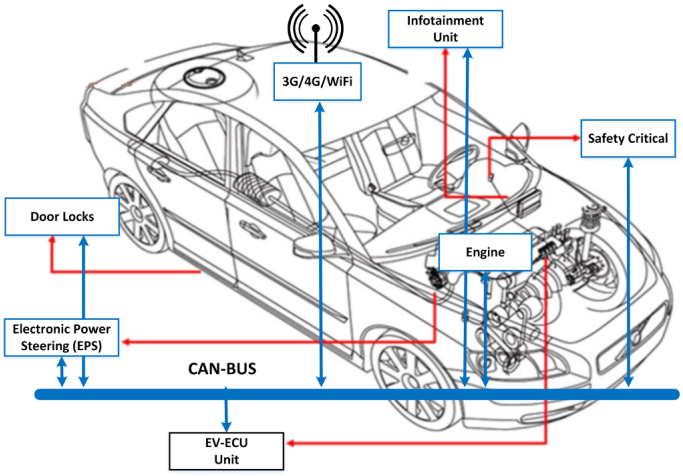

Fig. 2. Illustration of different system components of the connected car and their connectivity using CAN bus communication standard.

between controllers, actuators and sensors. Fig. 2 illustrates the connectivity of important car components (CAN nodes) that are connected via CAN bus which will be used in this case study. The internal architecture of a CAN node is shown in Fig. 3 that consists of a CAN transceiver, CAN controller and a micro-controller or digital signal processor (DSP). The CAN transceiver receives 2-wire differential CAN-H and CAN-L signals from CAN bus and converts them into single-ended digital signals. The CAN controller parse the received CAN-packet, to extract the control or status information of the connected sensors and actuators on the bus. By design, CAN is message-based broadcast communication protocol. Each connected CAN node can receive messages from any other node, which poses serious challenges to the security and safety of the car [24].

To cover wider scope of this case study, we envisage that the connected car features three operating modes, defined as car modes in the Table I, under which the vehicle's core functionalities will be adjusted:

1) Normal mode: Standard vehicle functionality and normal use cases, such as being driven or parked.
2) Remote Diagnostic mode: Reserved for maintenance by manufacturer or authorised engineer.

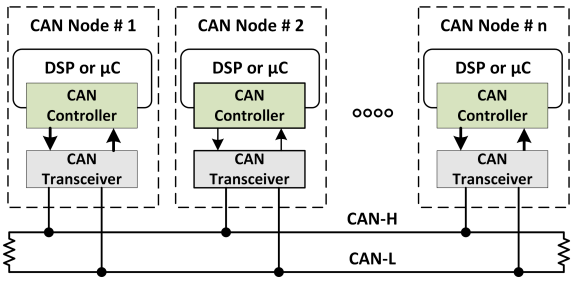

Fig. 3. Block diagram of multiple CAN nodes connected via shared CAN bus. Each CAN node has a dedicated CAN Transceiver, controller and a processor.

3) Fail-safe mode: Reserved for emergency situation.

For threat modelling, the car's chosen critical assets are *Electronic vehicle* (EV)-ECU, *electronic power steering* (EPS), Engine, 3G, 4G and WiFi, infotainment system, door locks and safety critical devices. For each mode, potential threats have been identified and their corresponding entry points are detailed in Table I. By employing threat analysis methods such as STRIDE to understand how the threat affects application and quantifying the damage of the realised threat using DREAD, it is possible to estimate the severity and likelihood the component will be targeted by an adversary. To address the threat, a policy will then be formulated. For the purposes of the example these will be *read* or *write* permissions, but more complex policies such as behavioural or situational based policies may be derived

A. Security and threat analysis of the EV-ECU asset

Three threats to the EV-ECU have been considered. In this example we will consider the sending of spoofed CAN bus data to cause disablement of the ECU during normal operation. In effect, this would cause the vehicle's propulsion mechanisms to become unresponsive. The only way to disable this mechanism under normal circumstances is when the car is locked and alarmed, when a safety critical event occurs such as an accident, or when a sensor detects an approaching object that requires immediate reaction, such as approaching a stationary object when parking. *Spoofing* or *tampering* of these components may therefore maliciously cause such reations during operation, resulting in a *denial of service*. This attack would likely cause damage to the vehicle's service ability, and affecting its users. However, such a threat would likely require specialist knowledge of the vehicle's ECU, the mentioned entry points and CAN bus. The reactive policy in this case is to permit only to read from the ECU.

1) Guideline based security specification: Under a normal security modelling approach, the described threat scenarios in Table I can be addressed using a guideline based security model. The countermeasures defined are guideline based. Further to the identified infotainment threat scenario, the following design guidelines could be used during development.

- Infotainment system: Provide frequent software updates and patch system when vulnerabilities are discovered.
- Infotainment system: Employ software protections to prevent unauthorised software installation
- CAN bus gateway: Limit components with CAN bus access

Due to being a guideline based approach, the OEM would have first have to redevelop the application or hardware with the new requirements integrated before deployment. In the worst case, a product recall may be necessary to install the proposed updates. Alternatively the OEM may reduce functionality through a software update, and integrate the fix within the next product cycle.

2) Policy-based security model: Our model proposes that new policies can be introduced, once a threat has been discovered. These can be implemented through a policy update,

TABLE I

THREAT MODELLING OF A CONNECTED CAR APPLICATION USE CASE. DIFFERENT THREATS TO THE IDENTIFIED CRITICAL ASSETS ARE COVERED TO DERIVE POLICY-BASED SECURITY MODEL OF THE APPLICATION.

Critical Assets	Car Mode: Normal	Car Mode: Remote Diagn.	Car Mode: Fail-safe	Entry Points	Potential Threats	STRIDE	DREAD (Avg.)	Policy
EV-ECU (accel, brake, transmission)	•			Door locks, safety critical	Spoofed data over CANbus causing disablement of ECU	STD	8,5,4,6,4 (5.4)	R
	•			Sensors		STD	8,5,4,6,4 (5.4)	R
		•		3G/4G/WiFi	Disabled remote tracking system after theft	SD	6,3,3,6,4 (4.4)	RW
			•	3G/4G/WiFi	Fail-safe protection override to reactivate vehicle	STE	5,5,5,7,6 (5.6)	R
EPS (Steering)	•			Any node	EPS deactivation through compromised CAN node.	STD	5,5,5,6,7 (5.6)	R
Engine	•			Sensors	Deactivation through compromised sensor	STD	6,5,4,7,5 (5.4)	R
3G/4G/WiFi	•			EV-ECU, Sensors	Critical component modification during operation	STIDE	7,5,5,9,4 (6.0)	R
		•		Infotainment system	Privacy attack using modified radio firmware	TIE	7,5,5,6,5 (5.6)	R
			•	Emergency, door locks	Prevent operation of fail-safe comms by disabling modem.	TDE	6,6,7,8,6 (6.6)	RW
			•	Sensors, Air bags		TDE	6,6,7,8,6 (6.6)	R
Infotainment System	•			Media player browser	Exploit to gain access to higher control level	STE	7,5,6,8,6 (6.4)	R
			•	Sensors, EV-ECU	Modification of car status values, GPS, speed, etc	STR	3,5,6,4,5 (4.6)	R
Door locks	•			3G/4G/WiFi, Manual open	Unlock attempt while in motion	TDE	8,5,3,8,5 (5.8)	R
			•	3G/4G/WiFi, Safety critical	Lock mechanism triggered during accident	TDE	8,6,7,8,5 (6.8)	W
Safety Critical	•			Sensors	False triggering of fail-safe mode to unlock vehicle	STE	7,4,5,8,4 (5.6)	R
			•	Sensors	Disable alarm and locking system to allow theft	TE	9,4,5,9,4 (6.2)	W

which would be significantly faster and easier to implement than a software redesign or product recall. The following examples are policies that may protect the device, in the context of the described infotainment scenario. Table I describes read and write policies that can be enforced at the entry points. More fine-grained policies may include the following;

- Infotainment system: Prevent software installation activities initiated from the media display
- Infotainment system: Enforce access of permitted commands using software-based policy method, eg SELinux
- CAN nodes: Enforce CAN ID verification on hardware policy engine at read/write filters within CAN controller

3) Advantages of the proposed approach: By utilising a policy-based approach, the OEM is not limited to addressing threats at the design phase. Upon detection of a new threat, policies can be defined to thwart the attack at a hardware or software level. These can then be distributed by a policy update mechanism. The entire cycle of threat and security modelling, along with implementation, testing and verification, prior to deployment, has potential to be much shorter and more effective than the standard guideline approach.

B. Policy enforcement approach

A vital component of our proposed policy-based security modelling approach is the enforcement of the driven policies within the device. This approach is inspired by the fundamental concepts of *least privilege* [25], which is enforced in computer systems through methods such as access control lists [26]. This section presents and discusses the possible software and hardware techniques that can be used to deploy policy-based approach:

1) SELinux-based policy enforcement: Policies are deployed using a modular approach, allowing administrators to tailor applications to their requirements [27]. Policies can be updated to apply new *Mandatory Access Controls* [28].

2) Hardware-based policy engine (HPE) architecture: Siddiqui *et al.* have presented a *hardware-based policy engine* (HPE), a system architecture that checks the approved list of devices and either grants or restrict access to the device [21].

Generally, the CAN node controller utilises a programmable software based filter. However, these may be vulnerable to software layer attacks, such as firmware modification. To this end, we propose a hardware-based policy engine that monitors the CAN messages and filters malicious and unexpected messages using their message IDs as shown in Fig. 4. The HPE consists of a separate hardware-based *reading filter* and *writing filter*, which facilitates curtailment of both inside (launched by

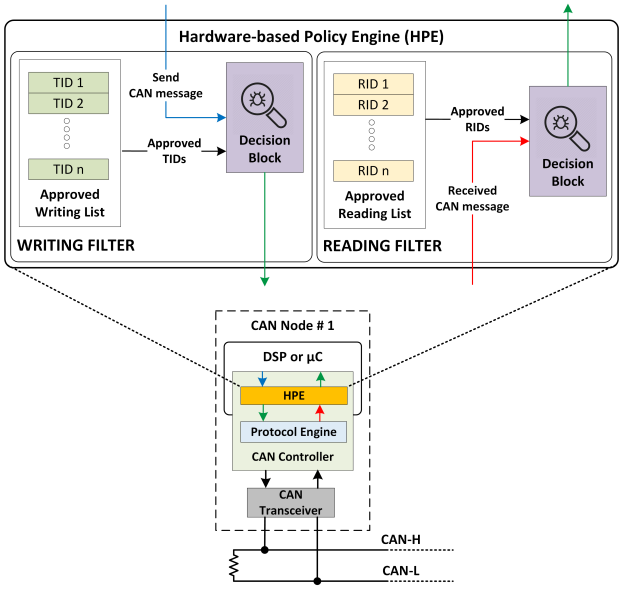

Fig. 4. Block diagram of CAN node with integrated hardware-based policy engine. It enforces security policies by filtering CAN messages.

a compromised node) and outside (launched by a malicious node introduced in the system to attack healthy nodes) attacks. In addition, the HPE provides an additional layer of defence over existing security mechanisms as it remain transparent to the system software. The HPE consists of following hardware components as shown in Fig. 4:

- *Approved reading and writing list:* It holds a list of approved CAN messages IDs that provides necessary information to the node to provide relevant services to the rest of the system without comprising the security.
- *Decision Block:* The decision block references the approved list of message IDs, compares it against the *issued/received message* and either grants or blocks the access as shown in Fig. 4.

Once the sanity check is completed, the CAN message can be either be used by the local CAN node (read) or send (write) it to the CAN bus which can be utilised by other nodes.

VI. Conclusion and future work

This paper's contribution has been to present existing approaches to device security modelling, presenting dificulties where they are restricted to influencing the design and development stage and not after deployment, when changes are more difficult to implement. This paper has therefore explored directly deriving and applying policies from the security modelling. This is particularly useful where the the vendor wishes to lower costs and provide a generic platform for multiple use cases, and where a flaw has been found in a device is already released in the market that could not be rectified without redesign.

This paper has walked through a realistic use case that shows how the application of policies can mitigate security flaws, with the alternative solution being a complex upgrade procedure that customers may be unwilling to perform.

Our proposed approach complements both existing work into threat models along with security modelling applications in software and hardware. The next phase of this research is to study the low level implementation of the proposed approach into a generic hardware platform and evaluate its effectiveness for systems with differing criticality.

References

[1] P. Spark, "White Paper: The route to a trillion devices: The outlook for IoT investment to 2035," ARM, Tech. Rep., 2017. [Online]. Available: https://community.arm.com/iot/b/blog/posts/white-paper-the-route-to-a-trillion-devices

[2] M. Antonakakis *et al.*, "Understanding the Mirai Botnet," in *26th USENIX Security Symposium (USENIX Security 17)*. Vancouver, BC: USENIX Association, 2017, pp. 1093–1110.

[3] E. Ronen, A. Shamir, A. O. Weingarten, and C. O'Flynn, "IoT Goes Nuclear: Creating a ZigBee Chain Reaction," in *Proc. IEEE Symposium on Security and Privacy (SP)*, May 2017, pp. 195–212.

[4] C. Valasek and C. Miller, "Remote Exploitation of an Unaltered Passenger Vehicle," 2015. [Online]. Available: https://www.defcon.org/html/defcon-23/dc-23-speakers.html

[5] S. Samtani *et al.*, "Identifying Supervisory Control and Data Acquisition (SCADA) Devices and their Vulnerabilities on the Internet of Things (IoT): A Text Mining Approach," *IEEE Intelligent Systems*, pp. 1–1, 2018.

[6] D. Lowry, E. Keary, and D. Rook, "Application Threat Modeling," OWASP, Tech. Rep., 2015. [Online]. Available: https://www.owasp.org/index.php/Application_Threat_Modeling

[7] Arm Limited, "ARM Platform Security Architecture Overview," ARM, Tech. Rep., 2017. [Online]. Available: https://pages.arm.com/PSA-Building-a-secure-IoT.html

[8] M. Wolf and D. Serpanos, "Safety and security in cyber-physical systems and internet-of-things systems," *Proceedings of the IEEE*, vol. 106, no. 1, pp. 9–20, Jan. 2018.

[9] H. Khattri, N. K. V. Mangipudi, and S. Mandujano, "HSDL: A Security Development Lifecycle for hardware technologies," in *Proc. IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, Jun. 2012, pp. 116–121.

[10] Cabinet Office, National security and intelligence, HM Treasury, and The Rt Hon Philip Hammond MP, "National Cyber Security Strategy 2016 to 2021," HM Government, UK, Tech. Rep., 2016. [Online]. Available: https://www.gov.uk/government/publications/national-cyber-security-strategy-2016-to-2021

[11] National Computer Security Center, *A Guide to Understanding Security Modeling in Trusted Systems*. DIANE Publishing Company, 1993.

[12] S. Chokhani, "Trusted products evaluation," *Commun. ACM*, vol. 35, no. 7, pp. 64–76, Jul. 1992.

[13] "Security enhanced linux," National Security Agency — Central Security Service, Tech. Rep., 2008. [Online]. Available: https://www.nsa.gov/what-we-do/research/selinux/

[14] P. Loscocco and S. Smalley, "Integrating flexible support for security policies into the linux operating system," in *Proceedings of the FREENIX Track: 2001 USENIX Annual Technical Conference*. Berkeley, CA, USA: USENIX Association, 2001, pp. 29–42.

[15] S. Ray, E. Peeters, M. M. Tehranipoor, and S. Bhunia, "System-on-chip platform security assurance: Architecture and validation," *Proceedings of the IEEE*, vol. 106, no. 1, pp. 21–37, Jan 2018.

[16] B. Dorsemaine *et al.*, "A new approach to investigate IoT threats based on a four layer model," in *Proc. IEEE 13th International Conference on New Technologies for Distributed Systems (NOTERE)*, Jul. 2016, pp. 1–6.

[17] R. Khan, K. McLaughlin, D. Laverty, and S. Sezer, "STRIDE-based threat modeling for cyber-physical systems," in *Proc. IEEE Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*, Sep. 2017, pp. 1–6.

[18] M. Cagnazzo, M. Hertlein, T. Holz, and N. Pohlmann, "Threat modeling for mobile health systems," in *Proc. IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, Apr. 2018, pp. 314–319.

[19] B. Tan, M. Biglari-Abhari, and Z. Salcic, "Towards decentralized system-level security for mpsoc-based embedded applications," *Journal of Systems Architecture*, vol. 80, pp. 41 – 55, 2017.

[20] N. Akatyev and J. I. James, "Evidence identification in iot networks based on threat assessment," *Future Generation Computer Systems*, 2017.

[21] F. Siddiqui, M. Hagan, and S. Sezer, "Embedded policing and policy enforcement approach for future secure iot technologies," in *Living in the Internet of Things: Cybersecurity of the IoT - 2018*, March 2018, pp. 1–10.

[22] E. Uhlemann, "Introducing Connected Vehicles [Connected Vehicles]," *IEEE Trans. Veh. Technol.*, vol. 10, no. 1, pp. 23–31, Mar. 2015.

[23] "Road vehicles – Controller area network (CAN) – Part 1: Data link layer and physical signalling," International Organization for Standardization, Geneva, CH, Standard, Dec. 2015.

[24] S. Checkoway *et al.*, "Comprehensive Experimental Analyses of Automotive Attack Surfaces," in *20th Proc. USENIX Conference on Security (SEC)*, ser. SEC'11, Aug. 2011, pp. 6–6.

[25] J. H. Saltzer, "Protection and the control of information sharing in multics," *ACM SIGOPS Operating Systems Review*, vol. 17, no. 7, pp. 388 – 402, Jul. 1973.

[26] R. S. Sandhu and P. Samarati, "Access control: principle and practice," *IEEE Commun. Mag.*, vol. 32, no. 9, pp. 40 – 48, Sep. 1994.

[27] M. Jahoda, B. Ančincová, and T. Čapek, "SELinux User's and Administrator's Guide," RedHat, Tech. Rep., 2018. [Online]. Available: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html-single/selinux_users_and_administrators_guide/index

[28] K. Macmillan *et al.*, "Design and Implementation of the SELinux Policy Management Server," in *Proc. of the Security Enhanced Linux Symposium*, 01 2006, pp. 1–6.